**Assignment20**

**1.**

class Node {

constructor(value) {

this.val = value;

this.left = null;

this.right = null;

}

}

function maxSubtreeSum(root) {

if (root === null) {

return 0;

}

const leftSum = maxSubtreeSum(root.left);

const rightSum = maxSubtreeSum(root.right);

return Math.max(

root.val + leftSum + rightSum,

leftSum,

rightSum

);

}

function maximumSubtreeSum(root) {

if (root === null) {

return 0;

}

const maxSum = maxSubtreeSum(root);

return maxSum;

}

// Example:

const root = new Node(1);

root.left = new Node(2);

root.right = new Node(3);

root.left.left = new Node(4);

root.left.right = new Node(5);

root.right.left = new Node(6);

root.right.right = new Node(7);

console.log(maximumSubtreeSum(root));

Output: 28

// Example usage:

const root = new Node(1);

root.left = new Node(-2);

root.right = new Node(3);

root.left.left = new Node(4);

root.left.right = new Node(5);

root.right.left = new Node(-6);

root.right.right = new Node(2);

console.log(maximumSubtreeSum(root));

Output: 7

**2.**

class Node {

constructor(value) {

this.val = value;

this.left = null;

this.right = null;

}

}

function constructBST(levelOrder) {

if (levelOrder.length === 0) {

return null;

}

const root = new Node(levelOrder[0]);

const queue = [root];

let i = 1;

while (queue.length > 0 && i < levelOrder.length) {

const node = queue.shift();

const leftVal = levelOrder[i++];

if (leftVal !== null && leftVal !== undefined) {

node.left = new Node(leftVal);

queue.push(node.left);

}

const rightVal = levelOrder[i++];

if (rightVal !== null && rightVal !== undefined) {

node.right = new Node(rightVal);

queue.push(node.right);

}

}

return root;

}

// Example:

const levelOrder = [7, 4, 12, 3, 6, 8, 1, 5, 10];

const root = constructBST(levelOrder);

function inOrderTraversal(node) {

if (node === null) {

return;

}

inOrderTraversal(node.left);

console.log(node.val);

inOrderTraversal(node.right);

}

inOrderTraversal(root);

**3.**

function canRepresentBST(arr) {

const n = arr.length;

let i = 1;

while (i < n) {

if (arr[i] > arr[0]) {

break;

}

i++;

}

while (i < n) {

if (arr[i] < arr[0]) {

return "No";

}

i++;

}

return "Yes";

}

// Example:

const arr1 = [7, 4, 12, 3, 6, 8, 1, 5, 10];

console.log(canRepresentBST(arr1)); // Output: Yes

const arr2 = [11, 6, 13, 5, 12, 10];

console.log(canRepresentBST(arr2));

Output: No